A graph representation of gapped patterns in phage sequences for graph convolutional network

Ruohan WANG[1], Yen Kaow NG[1], Xianglilan ZHANG[2], Jianping WANG[1,*], and Shuai Cheng LI[1,*]

[1]Department of Computer Science, City University of Hong Kong, Hong Kong, 999077, China
[2]State Key Laboratory of Pathogen and Biosecurity, Beijing Institute of Microbiology and Epidemiology, Beijing, China
[*]jianwang@cityu.edu.hk; shuaicli@cityu.edu.hk

ABSTRACT

Phage mining yields a large number of phage genome sequences. There is an urgent need for tools to analyze the obtained phage sequences reliably. Neural network-based methods are prime candidates for these tasks due to their applicability to large, diverse datasets where little knowledge is available. However, the highly variable lengths of nucleic acid sequences pose a major obstacle to the sequence presentation as neural network input. Also, genetic variations further complicate the sequence comparison. Here we propose a graph representation of nucleic acid sequence, where all subsequences of the represented sequence can be matched to gapped patterns formed by paths of the graph, and various distribution information of these subsequences are encoded as features of the graph elements. These graphs, which we call *gapped pattern graphs*, can be transformed through a Graph Convolutional Network (GCN) to form lower-dimensional embedding for downstream tasks. Based on the gapped pattern graph, we introduced Graphage, an implementation of a neural network model, and compared it with equivalent models based on other forms of input (e.g., k-mer distribution, word2vec) in performing four downstream tasks: phage and ICE discrimination, phage integration site prediction, phage lifestyle prediction, and phage host prediction. We also compared Graphage with the current state-of-the-art tools on these tasks, where such tools are available. Our results show that Graphage consistently outperformed all the other tools and methods under various metrics on all four tasks. With Graphage, we identified distinct gapped pattern signatures with regard to phage phenotypes.

Keywords: Graph convolutional network; Representation learning; Phage analysis

Introduction

Viruses infecting bacteria or archaea (i.e., bacteriophages or phages) are the most abundant and diverse biological entities on Earth[1–3]. Phages are classified into virulent and temperate phages according to their lifestyles. Virulent phages release virion and destroy host cells upon infection (lytic cycle). Temperate phages can integrate into host genomes as prophages and reproduce with host replication (lysogenic cycles); they can also enter lytic cycles under stress conditions[4,5]. Through lytic or lysogenic infection, phages play essential roles in maintaining species diversity and drive bacterial co-evolution[6,7]. Additionally, with the threat of multi-drug resistance, phage therapy, which uses phages to treat bacterial infections, is being looked upon as an alternative to traditional antibiotic therapy[8,9]. However, despite the significant impact on microbial ecosystems and clinical value, our knowledge of phages remains limited due to difficulties in laboratory cultivation[10].

Advances in next-generation sequencing (NGS) and phage detection methods have enabled many phage sequences extracted or mined computationally from bacterial and metagenomic NGS data[11–13]. These methods detect phage-related features or signals from the assembled data and recover the sequences of the phage genome[14–17]; they allow phage discovery without culturing, thus enabling very rapid expansion in the diversity of known phages. On the other hand, new phage sequences obtained from NGS data require substantial work to analyze.

There are several shortcomings in NGS-derived phage sequences that complicate their analyses. First, integrative and conjugative elements (ICE) have homological genes and similar lifestyles to phages; they may contaminate NGS data[12] and affect phage sequence reliability. Second, boundary precision in *in silico* predicted prophages is often dissatisfactory, which limits downstream analyses such as genome size statistics and integration mechanism study[18]. Third, in many situations (e.g., metagenomics data), phage mining cannot provide some essential phenotypes, such as lifestyle and host[19,20]. The lack of such information impedes phage-host interaction study[21] and the application of phage therapy[22].

In addition to these problems, NGS-derived phage sequences require characterization, which is inherently a problem of homology with known sequences. However, the runtime complexity of the homology search is highly dependent on

the database size, which is overwhelmingly large for phage sequences. To avoid time-consuming alignment in sequence comparison, homology in phage analysis has been computed with alignment-free (AF) methods[23–25]. These methods calculate the frequency or uniqueness of different k-mers to characterize genomes, and use various statistical measures to determine the similarity between different phage genomes. However, several drawbacks limit the robustness of these AF approaches. First, approaches based solely on k-mer distributions forfeit all possibility of capturing critical information in the order or interaction of k-mers. Second, most AF approaches require exact matches of k-mers, which overlook the frequent genetic variations in phage genomes[26], including single nucleotide polymorphisms (SNPs) and insertions or deletions (InDel).

In recent years, there has been an avalanche of neural network methods for bio-sequence analysis. Most of these methods rely on one-hot encoding of sequences as input. Such an encoding adapts poorly to phage sequences due to the variation in phage sequence lengths and has no capability to associate neighboring nucleotides into components[27]. As a result, k-mer frequency vectors are commonly used in phage analysis[12,28]. However, using an AF-based input entails all of the disadvantages mentioned earlier. An additional drawback restricts all such vector-based inputs of k-mer information — they impose an implied ordering on the k-mers since each k-mer is identified by its position within the vector, thus introducing a bias on the neural network models. On the other hand, a nonarbitrary ordering of the k-mers is challenging to realize. For instance, one can require neighboring k-mers to share the exact substrings. However, since a k-mer has $3k$ neighbors that differ by a single letter, they cannot be arranged in matrices. A simpler and natural solution to these drawbacks is to represent each k-mer as a vertex in a graph. This way, there would be no imposed ordering on the k-mers (i.e., vertices), and interactions between k-mers can now be modeled as edges in the graph.

We call such a graph a *gapped pattern graph* (GPG) and construct it as follows. Given a phage sequence, we create a vertex to represent each k-mer in the sequence; the vertex is assigned a feature that encodes the k-mers' frequencies within the sequence. An edge connects two vertices if their corresponding k-mers appear within proximity at least once in the sequence. The construction hence (1) encodes the k-mer distribution; and (2) enables a walk of the graph to represent a potential pattern (in the form of a few k-mers with gaps in-between) within the sequence. These k-mers with gaps attempt to resemble the patterns that contain *spaced seeds*[29], which are known to tolerate minor substitutions, InDels, sequence errors, *etc.*.

The comparison of two graphs (i.e., phage sequences) can be naturally performed through so-called graph neural networks[30], which have been studied very intensely in recent years. We train a Graph Convolutional Network (GCN)[31] that transforms each GPG into a vector in a low-dimensional latent space; the vectors are then used in downstream phage analysis tasks. We call our resultant framework *gapped-pattern-GCN* (GP-GCN). Based on the framework, we developed Graphage, a tool that performs four phage-related tasks: phage and ICE discrimination, phage integration site prediction, phage lifestyle prediction, and phage host prediction. To our knowledge, the framework is the first method utilizing GCN to explore pattern relationships in biological sequences.

Results

Overview of Graphage

In this study we present Graphage (Figure 1), an implementation of the GP-GCN framework. In the input preparation stage, each phage sequence S is converted into a GPG (see *Gapped Pattern Graph Construction* subsection). Each of the sequence's constituent k-mers becomes a vertex, and is assigned the k-mer's frequency as feature. An edge points from a vertex of the k-mer s to a vertex of the k-mer t if and only if t occurs after s at least once in S within some threshold distance. Hence such an edge represents a pattern of the form sxt, where s and t are k-mers and x is an arbitrary string of up to a given length. The edge is assigned the frequency of the pattern sxt in S as feature.

The first part of Graphage transforms each GPG into a low-dimensional embedding (latent space vectors) using a multilayer GCN (see *Neural Network Architecture* subsection) (Figure 1A). The second part implements plain (fully-connected) neural network models for prediction tasks (Figure 1B). These models accept the embeddings as input and output predictions. Four neural network models are used, each corresponding to a specific phage analysis task. The GCN and the downstream model are trained concurrently.

Graphage is available at `https://github.com/deepomicslab/Graphage`. We also provide the framework at `https://github.com/deepomicslab/GCNFrame`.

Phage and ICE discrimination

For phage and ICE discrimination, we use the sequences from NCBI[32] and ICE[33] as training set. A classifier is used to distinguish the two classes from the embeddings. Following the practice introduced in GPD[12], in addition to the embeddings, we include gene density and hypothetical protein fraction information as input to the classifer. We optimized two hyperparameters: the maximum allowed gap length (d) and the number of graph convolutional layers (l), through grid search. In our tests $d = 1$ and $l = 3$ achieved the best performance (Supplementary Figure S1). More hyperparameter settings can be found in Supplementary Table S1.

When tested on a separate database ImmeDB[34], Graphage outperformed AF-based, word2vec-based, and sequence-based models, with 7.26% increase in average accuracy, 0.1117 increase in F1-score, and 0.0181 increase in average Area under ROC (AUC), compared to the respective second highest scores (Table 1). The sequence-based model gave the worst performance, consistent with our argument that the method is unsuitable for sequences of variable lengths. For reference we also performed phage and ICE discrimination with GPD[12]. We found that the GPD model is likely to misclassify ICEs as phages, leading to lower accuracies and F1 scores.

Since a GPG contains information of both the frequencies of k-mers (vertex feature) and gapped patterns of two k-mers (edge feature), it is worth questioning if the latter could be solely responsible for the performance of Graphage. To answer this question we performed tests with the frequencies of the gapped patterns (and genetic features) as input to several conventional machine learning models (see Competing Methods subsection). The results show that this is not the case (Table 1).

It could be the case that Graphage's classifier depended only on the embeddings, or only on the genetic features (i.e., gene density and hypothetical protein fraction) for its output. To evaluate this possibility we re-evaluated Graphage with only the embeddings or genetic features as input. We found that the best performance is achieved with both inputs, showing that they jointly contributed to the performance (Supplementary Figure S2).

Phage integration site prediction

Phage integration site prediction is complicated by the fact that circular phage genomes can integrate into the bacterial genome as prophages. To distinguish them we trained two models that respectively predict phage integration sites on phage genomes and bacterial genomes. A dataset is prepared for each model. The phage integration site dataset consists of 21,117 positive samples and 21,117 negative samples; the bacterial integration site dataset consists of 17,550 positive samples and 17,550 negative samples. We used 90% of the data for training and validation. Besides the allowed gap length (d) and the number of graph convolutional layers (l), we also optimized the window length (w), i.e., the input sequence length. The results show that $d = 2$, $l = 3$, and $w = 600$ gave the best performance for integration site prediction for both models (Supplementary Figure S3 and Figure S4). More hyperparameter settings are shown in Supplementary Table S2.

When evaluated on the testing sets, both models obtained prediction accuracies higher than 0.8, significantly outperforming AF-based, word2vec-based, and sequence-based models under various metrics (Tables 2 and 3). For integration site prediction on phage sequence, Graphage has a 3.90% lead on average accuracy over the runner-up's score; this lead is 2.80% on bacteria genomes. We observe that the sequence-based model performs better when inputs are of the same length than when they are of variable length.

Since a GPG contains information of both the frequencies of k-mers (vertex feature) and gapped patterns of two k-mers (edge feature), we performed tests with the latter as input to several conventional machine learning models (Tables 2 and 3).

Phage lifestyle prediction

In lifestyle prediction, we want to classify phages by whether they are virulent or temperate. Since sequences alone are likely insufficient for this classification, we supplemented the classifier with additional information by using a dataset that contains 206 lysogeny-associated proteins as established by BACPHLIP[20]. We used HMMER[35] to check the presence of the 206 proteins in each sequence and encoded the information into a 206-dimensional vector, with 0 for absence and 1 for presence. The maximum allowed gap length (d) and the number of graph convolutional layers (l) were optimized through grid search. The model with $d = 2$ and $l = 4$ achieved the best classification performance (Supplementary Figure S5). More hyperparameters can be found in Supplementary Table S3.

Graphage gave the highest average performance among all competing methods, including AF-based, word2vec-based, and sequence-based models, as well as two phage lifestyle prediction tools, BACPHLIP[20] and DeePhage[36] (Table 4). Compared to the other methods, Graphage demonstrated large improvements in accuracies and F1-score metrics. We also examined the performance achievable by using gapped patterns of two k-mers (and protein information), through using them as input to several conventional machine learning models (Table 4).

To determine the contribution from the additional information, we trained two Graphage models, one with only embeddings and one with only lysogeny-associated protein information. While both models performed poorer than that with both information (Supplementary Figure S6), the model with only protein information had an edge over that with only embeddings.

Phage host prediction

Phage host prediction is a multiclass prediction problem, where there are 107 host species in our constructed datasets. We use 90% of the data for training and validation. Accuracy, weighted F1-score, and macro F1-score were used as evaluation metrics (see METHODS). Under these metrics, the model with $d = 2$ and $l = 1$ achieved the best performance (Supplementary Figure S7). More hyperparameter settings are shown in Supplementary Table S4.

In addition to AF-based, word2vec-based, and sequence-based, we also evaluated the performance of HostPhinder[37], VirHostMatcher[38], WIsH[38] and DeepHost[39]. Graphage outperformed all the other methods (Table 5), except for AF-based and

KNN-based models, which achieved comparable performance to Graphage in both accuracy and weighted F1 score. These methods are based on the principle of nearest neighbors, which generalizes well to multiclass prediction tasks. In macro F1-score, Graphage outperformed all other methods. With the exception of VirHostMatcher, all methods scored much worse in macro F1-score compared to weighted F1-score, since predicting the samples from rare classes is much more challenging. Finally, we showed gapped patterns of two *k*-mers to be insufficient for good performance (Table 5).

In the tests using VirHostMatcher and WIsH, we constructed the bacterial database using the references of the 107 bacteria. These methods output the bacteria with the most similar sequence with the phage as their prediction. Both methods have no phage datasets with known hosts, which could have led to a compromise in their performance.

Necessity of GCN component

To verify the necessity of the GCN component within the GP-GCN framework, we performed an ablation study by removing the GCN while keeping other components the same. We trained new models for the four tasks and compared the performance with Graphage. The results, given in Supplementary Table S5, show that the models with GCN achieved superior performance on all four tasks.

We further visually inspect the influence of GCN on the final output of Graphage on a binary classification task. More precisely, we examine the outputs of Graphage right before its final, fully-connected layers, to see how well the outputs of sequences from the same class form clusters; we compare the clusters for the case when GCN is used and when it is not. To enable visualization we use uniform manifold approximation and projection (UMAP)[40] to project the output values for each sequence into two dimensions. As shown in Figure 2, the outputs when GCN is used form more clearly defined clusters in accordance with their labels.

Mining gapped patterns of greater influence in prediction task

As mentioned, every edge of a GPG represents a gapped *k*-mer pair pattern. It is possible that some of these gapped patterns may exert more influence in a prediction task than others. To investigate this possibility, for a given input sequence, we remove the feature of an edge from the sequence's GPG and examine how that influences the prediction accuracy of a trained Graphage model on the sequence. More precisely, given a phage sequence S, we first obtain the trained Graphage output probability for the class that S belongs to. Then, we obtain the GPG of S and set the feature of a given edge in the GPG to zero and obtain the new corresponding Graphage probability; the difference between the two probabilities gives us the influence of the gapped pattern that corresponds to the given edge. This is repeated for every sequence, and for every edge of the GPG of the sequence. The total contribution of each gapped pattern is aggregated from all the differences resulting from its edge being set to zero.

We show the gapped patterns with contribution scores higher than 0.1 for phage lifestyle prediction in Supplementary Figure S8 A. The occurrence frequencies for the five gapped patterns with the highest contribution scores are given in Supplementary Figure S8 B; frequencies for the five with the lowest contribution scores are shown in Supplementary Figure S8 C. We note that the gapped patterns with the highest contribution scores demonstrated significantly different frequencies between temperate phages and virulent phages, suggesting their significance in the classification task.

We failed to observe gapped patterns with attribution scores higher than 0.1 in the other three prediction tasks (Supplementary Figure S9). We suspect that the models for these tasks combine the information from several gapped patterns in computing their outputs, making the significance of single gapped pattern unremarkable. Due to the rapidly increasing time complexity we do not perform tests with combinations of multiple patterns.

Regulatory motifs involved in phage integration

To study the phage integration mechanism, we looked into the involved regulatory motifs. First, we applied STREME[41] to discover the motifs enriched in the sequences of the integration site with the randomly chosen negative sequences as control. As a result, we obtained 256 motifs from phage integration sites and 243 motifs from bacterial integration sites. Next, we utilized Tomtom[42] to compare these motifs against the CollecTF dataset[43], which collects the transcription factor (TF)-binding sites in Bacteria. Setting the *p*-value threshold as 0.01 and the percentage of sequences matching the motifs threshold as 20%, we identified 26 motifs from phage integration sites and 16 motifs from bacterial integration sites, which have significant matches with the TF-binding site patterns in CollecTF.

In Figure 3 A and C, we show the averaged contribution score and the alignment *p*-value for these motifs. Graphage computes the averaged contribution score of a motif by removing all the gapped patterns in the motif from the sequences and calculating the reduced output probability on average. For the eight motifs with the largest number of matching sequence, we present the sequence logo alignment (Figure 3 B and D). Specifically, for the phage integration site (Figure 3 B), the frequently observed motif "ACTADWYA" matches with LexA. LexA was reported to regulate the genes on mobile genetic elements for their coexistence with the hosts[44]. Also, the motif "AATTKGT" can be aligned to LuxR, which is encoded by phages and involved in phage-host communication[45]. For the bacterial integration site (Figure 3 D), the motif "AMAAMWCA" matches

with ToxT, which is found in the region containing a putative integrase[46]. The motif “TATAATTT" is consistent with PerR, which is required for the expression of phage-associated DNase[47].

Robustness of GCN-produced embeddings

We evaluated the robustness of the GCN-produced embeddings through large-scale simulated sequences. Two criteria were used to evaluate the embeddings: 1) similar sequences should have similar embeddings and vice versa, 2) embeddings should remain the same under a reasonable amount of genomic variations. As dataset we randomly selected 100 phage sequences from NCBI and simulated their genomic variants, using insertion, deletion, inversion, and translocation. We generated three datasets with different variation rates and lengths (See Supplementary Method S1.2 for construction details of the simulated dataset).

To examine the first criterion, we obtained embeddings using the GCN component of the model trained for the phage and ICE discrimination task. The dissimilarity between two embeddings is computed as the Euclidean distance between the embeddings. As shown in Supplementary Figure S10, distances between the embeddings are small even with large variations in the simulated sequences. We compared these distances to the case where the GCN is replaced with a CNN that embeds sequences into vectors of the same dimension as the GCN-produced embeddings. For all of the three datasets, GCN gave embeddings with smaller distances between the original and the mutated sequences.

Scalability of the GP-GCN framework

Due to the very large number of biological sequences in phage analysis, any method for analyzing phage sequences must be capable of handling millions of sequences. The GP-GCN encoding framework is learning-based; after the initial training, the model can perform predictions in runtime that is linear to the number of sequences to infer. Additionally, each gapped pattern graph can be constructed in time polynomial to the sequence length (see Supplementary Method S1.3). The computational efficiency makes the framework suitable for handling large datasets.

Discussion

The phage genomes are known for their frequent genetic variations, which complicates any analysis that requires a homology search. Current computations of homology are mainly alignment- or AF-based. Alignment-based methods face runtime complexity issues, while AF-based methods compromise important information in the sequences. As phage sequence databases become large and sequence diversity increases, such compromises are beginning to limit phage genome analyses.

In this work, we propose a novel method for succinct biosequence representation, where each sequence is represented as a *k*-mer-encoding graph called GPG. To enable efficient comparison of these GPGs we exploit recent advances in graph neural networks to filter such graphs into low-dimensional embeddings. These ideas give rise to a framework which we call GP-GCN. Based on the framework, we developed Graphage, a phage analysis tool currently performing four tasks: phage and ICE discrimination, phage integration site prediction, phage lifestyle prediction, and phage host prediction. Graphage achieves state-of-the-art performance compared to other methods on all four tasks.

The application of representation learning to biological sequences is not new. For instance, word2vec, a widely used natural language processing (NLP) technique[48,49], has been applied to obtain embeddings from sequences of the human genome[50], as well as solve problems such as species identification[51,52], methyladenosine site prediction[53,54], and MHC binding site prediction[55,56]. However, word2vec is based on the encoding of local contextual information of sequence segments and have no provision for representing global information of entire sequences. In our tests, Graphage consistently outperformed word2vec-based models.

We expect the GP-GCN framework to be useful beyond the prediction tasks currently implemented in Graphage due to its many desirable properties, such as robustness with respect to genomic variations and high computational efficiency. Our immediate plan is to attempt the framework on more types of sequences, e.g., RNA and protein sequence, as well as on more complex downstream tasks, such as sequence generation or sequence restoration. One application, clustering of genome sequences, is of fundamental importance to many studies such as phylogenetics and has spurred the development of many tools[57–60]. The ability to cluster biosequences more accurately and in larger scales would benefit many researchers.

Other possibilities abound for the practical use of the GP-GCN framework. For instance, trained GP-GCN models encode important domain knowledge of the tasks they were trained for. Mining gapped patterns from the models trained in the four tasks in Graphage has allowed us to uncovered patterns that are possibly relevant to the phage lifestyle. Another possible use of the framework is in the discovery of partial similarities between GPGs. This may be useful in situations where the GPGs are constructed from genetic materials consisting of multiple species, such as in metagenomic studies. In such a case, partial GPG similarity would suggest a similar sub-composition of the two inputs.

Methods

Benchmark datasets

We collected seven published datasets for the four phage-related tasks (Table 6, Figure 4). For phage and ICE discrimination, following the data preparation process of GPD[12], we used the phage genomes from NCBI[32] and ICE[33] sequences from ICEBerg for training. Then we downloaded an independent dataset, ImmeDB[34], including ICE and prophage sequences, for test. We applied Prokka[61] for sequence annotation to obtain the information of gene density and hypothetical protein fraction. For phage integration site prediction, we applied a temperate phage dataset with precise boundary information[13]. We extracted the sequences of integration sites from both phage and bacterial genomes as positive sequences and randomly chose negative sequences 1 *kb* away from the integration site on phage and bacteria genomes with the same length. For phage lifestyle prediction, we downloaded the phage sequences with empirical lifestyle data[62] and tagged them with the lifestyle-related protein domains selected by BACPHLIP[20]. We applied Glimmer[63] to identify the genes in phage genomes and HMMER[35] to search for the protein domains. For phage host prediction, we combined one dataset mined from metagenomic data[12] and one dataset mined from bacterial data[13] with their host information. We removed the phages with unidentified hosts and kept only the taxonomies with more than 20 phage genomes. All redundant sequences were removed from each of the seven datasets.

Gapped pattern graph construction

Notation and definition

For a given phage sequence S of the alphabet $\{A,G,T,C\}$ and a positive integer k, the k-mers of S are all the substrings of S of length k. Examples of 2-mers are AA and GA, while examples of 3-mers are ACG and GAG. There are a total of 4^k distinct k-mers for any positive integer k, and at most $\min\{4^k, d-k+1\}$ unique k-mers can occur in a sequence of length d ($d \geq k$). An arbitrary indexing can be imposed over the set of all k-mers for any fixed k. Such an indexing can be used as indices in a vector, thus allowing the k-mer distribution of a sequence to be represented as a vector of integers.

Pattern graphs

Given positive integer k, a *pattern graph* $G(S,k)$ of a sequence S is a directed graph where each vertex corresponds to a k-mer of S. For two vertices $u,v \in G(S,k)$, an edge from u to v is in the graph if and only if the concatenation of the two k-mers that the vertices correspond to, S_u and S_v say, exist in S. For example, $G(ATGATGC,3)$ would consists of four vertices v_1,v_2,v_3,v_4 (corresponding respectively to the k-mers ATG, TGA, GAT, and TGC), and have exactly two edges: the edges $v_1 \to v_1$ and $v_2 \to v_4$. Each vertex in $G(S,k)$ is assigned a number which indicates how frequently its corresponding k-mer occurs in S; similarly, each edge is assigned a number indicating how frequently the concatenation of the two k-mers occurs in S.

Two notions of frequencies are considered for the vertex and edge features: absolute or normalized counts. An absolute count of a k-mer is the number of times the k-mer occur in S; the normalized count of a k-mer is the number of times the k-mer occur in S divided by the total number of k-mer occurrences (i.e., $L-k+1$). These variants of frequency are similarly defined for the edge features. In this work, we use normalized count for both vertex and edge features.

Supplementary Figure S11 shows examples of this construction.

Gapped pattern graphs

To allow SNPs and InDels in a phage sequence, we construct *gapped pattern graph* (GPG) as follows. Let d be the maximum allowed gap length. The GPG $G(S,k,d)$ is defined in the same way as the pattern graph, except for the edges. For two vertices $u,v \in G(S,k,d)$, an edge from u to v is in the graph if and only if the k-mers that the vertices correspond to, S_u and S_v say, exist in S with S_v occurring within distance d after S_u. That is, $S_u x S_v$ is a substring of S, where x is an arbitrary (possibly empty) string of length at most d. Each edge is assigned a vector of length $d+1$, the $(i+1)$-th element of which indicates the frequency of $S_u y S_v$ in S, where y is an arbitrary string of length i. Varying the parameter d allows more flexibility in the gap length, hence increasing the expressiveness of the pattern space spanned by the GPG. Examples of this construction are given in Supplementary Figure S12.

Just as every edge in a GPG can be considered a pattern of two k-mers with a gap of up to length d in between, every walk of length l on a GPG can be considered a pattern that consists of $l+1$ k-mers, with gaps of up to length d in between. Given the GPG of a sequence S, all subsequences of S can be matched to one or more gapped patterns formed by such walks. Note that the other direction of this statement is not necessarily true for patterns with more than two k-mers.

In this work we typically let $k=3$ and $d=2$ (See Supplementary Tables S1-S4). However, for large values of d, we can compress the vector by letting some vector elements account for a range of gap lengths. For instance, we can let the first three elements indicate the frequencies of $S_u y S_v$ where y has lengths of 0, 1, and 2 respectively, but let the fourth element indicate the frequencies where y has lengths of 3 or 4, the fifth element indicate the frequencies where y has lengths from 5 to 9, and so on.

Implementation details

Graphage uses GCN to transform each GPG into an embedding. However, current software libraries for GCP treat edges and vertices differently, whereas our formulation prefers that they be treated equally. In particular, we want edge features to be modified in the same way as vertex features.

To overcome this limitation, we insert a vertex on every edge in the GPG and assign the feature of the edge to the new vertex. The new vertex maintains the connections of the original edge; hence every edge becomes a new vertex of in-degree 1 and out-degree 1. We call these new vertices *k-mer pair vertices* to distinguish them from the *k-mer vertices* in the original GPG. This conversion does not change the information flow of the original GPG; on the other hand, all the features on the GPG are now associated with vertices, thus allowing their manipulation using standard GCN libraries.

From the construction, each k-mer vertex is only connected to k-mer pair vertices, and vice versa. An edge is set to point from a k-mer vertex to a k-mer pair vertex, then back to a k-mer vertex. An example of this construction is given in Supplementary Figure S13.

Neural network architecture

After constructing the GPGs from phage sequences, a GCN is used to transform each GPG into an embedding. We use the PyTorch Geometric[64] implementation of GraphSAGE[65] for GCN (i.e., SAGEConv). Using the default settings, each graph convolutional layer computes the function,

$$h'_v = \sigma\left(Bh_v + W \cdot \mathsf{mean}(\{h_u | u \in N(v)\})\right) \tag{1}$$

where h_v is the feature of vertex v, $N(v)$ is the set of all vertices in the graph where edges incident to v emanate from, σ is a non-linear activation function, mean computes the mean of the input features, and W, B are matrices to be learned. (Since GPGs are directed graphs, the use of such a function implies a spatial paradigm rather than a spectral one. Spectral GCN models for directed graphs exist[66–68] but they are not yet generally available in standard libraries.)

As discussed in *Implementation Details* above, we convert edge features in the GPG into vertex features. Since the feature length of the k-mer vertices does not match that of the k-mer pair vertices in general, we use a two-step feature computation as follows. Denote a GPG as $G(S,k,d) = (V,U,E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is the set of k-mer pair vertices, $U = \{u_1, u_2, \ldots, u_{n'}\}$ is the set of k-mer vertices, and $E = \{(v,u) | v \in V, u \in U\} \cup \{(u,v) | v \in V, u \in U\}$ is the set of edges. The input features for k-mer pair vertices $X_v = \{x_v | v \in V\}$ and input features for k-mer vertices $X_u = \{x_u | u \in U\}$ are the encoded frequencies, where $x_v \in \mathbb{R}^{d+1}$ and $x_u \in \mathbb{R}$. A graph convolutional layer at level l accepts as input a list of k-mer pair vertex features $H_v^l = \{h_{v_1}^l, h_{v_2}^l, \ldots, h_{v_n}^l\}$ and a list of k-mer vertex features $H_u^l = \{h_{u_1}^l, h_{u_2}^l, \ldots, h_{u_{n'}}^l\}$. Clearly, $h_v^0 = x_v$ and $h_u^0 = x_u$. The layer outputs $H_v^{l+1} = \{h_{v_1}^{l+1}, \ldots, h_{v_n}^{l+1}\}$ and $H_u^{l+1} = \{h_{u_1}^{l+1}, \ldots, h_{u_{n'}}^{l+1}\}$ which are computed as

$$h_u^{l+1} = \sigma\left(B_1^l h_u^l + W_1^l \cdot \mathsf{mean}(\{h_v^l | v \in N(u)\})\right) \tag{2}$$

$$h_v^{l+1} = \sigma\left(B_2^l h_v^l + W_2^l \cdot \mathsf{mean}\left(\{h_u^{l+1} | u \in N(v)\}\right)\right) \tag{3}$$

where W_1^l, W_2^l, B_1^l and B_2^l are matrices to be learned. The outputs of the graph convolutional layer at level l is given as inputs to the graph convolutional layer at level $l+1$. Note that we use h_u^{l+1} instead of h_u^l in the computation of h_v^{l+1} (Eqn 3) as a matter of preference.

The output of the final graph convolutional layer is concatenated and given as input to a CNN[69] for dimensionality reduction to produce embeddings that are used in downstream tasks.

In the present work, all downstream tasks are constructed as simple fully connected layers that perform classification. For the phage and ICE discrimination task and the phage lifestyle prediction task, we give the genetic features to the first fully-connected layer.

Training process and hyperparameters

We split data into training, validation, and test data (See Supplementary Table S1-S4 for details). During training, we use weighted sampling to overcome label imbalance in the training data. The sampling weight of a label is set to the inverse of its frequency in the training data. We apply mini-batch gradient descent to optimize the cross-entropy[70] loss function

$$L = -\frac{1}{N_b} \sum_{(x,y)} \sum_{c=0}^{K-1} y_c \log p_c(x), \qquad y_c = \begin{cases} 1, & \text{if } y = c \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

where N_b is the batch size, x the network input, y the label, K the number of classes, and $p_c(x)$ the predicted probability that x belongs to class c. The Adam gradient descent algorithm[71] is used in back-propagation with a learning rate of 10^{-4}.

The neural networks are implemented in PyTorch[72] and PyTorch Geometric[64]. Other hyperparameters and settings are summarized in Supplementary Table S1-S4. All experiments are conducted on an Nvidia Tesla T4 (16G).

Competing methods

To evaluate the performance of our trained models, we compared Graphage against AF-based (i.e., k-mer distribution), word2vec-based, and sequence-based models of similar constructs as follows. (Since we use GPGs of 3-mers, the resultant GPGs encode frequencies of 4^3 3-mers, and of up to 4^6 gapped 3-mer pair patterns. Hence, we provide the distribution of 4^6 6-mers, i.e., a roughly equivalent amount of information, as input to AF- and word2vec-based models.)

- The AF-based models are written using alfpy[73]. We represent each sequence with a 6-mer frequency vector, then calculate the distances between the sequences in the training set and the sequences in the test set. Prediction for a test sequence is performed by returning the label of its nearest neighbor in the training set.
- For word2vec, we apply Gensim[74] to learn the 6-mer vector representations from the training sequences. Then a pooling strategy over the 6-mers is applied, and an optimized fully-connected neural network is used for the final prediction.
- For sequence-based methods every sequence is converted into a vector of a fixed length. More precisely, each sequence is straightforwardly transformed into a matrix of shape $4 * L$ with one-hot encoding (A is encoded by $(1,0,0,0)$; C is encoded by $(0,1,0,0)$; G is encoded by $(0,0,1,0)$; T is encoded by $(0,0,0,1)$). For phage and ICE discrimination, phage lifestyle prediction, and phage host prediction, we set L to be 50,000, which is the median length of phage sequences. Sequences shorter than 50 *kb* are padded with trailing zeros, while sequences longer than 50 *kb* are discarded. An optimized fully-connected neural network is used for the final prediction.

We furthermore compared Graphage to other state-of-the-art tools, as follows:

- For phage and ICE discrimination, GPD[12] was used for comparison.
- For phage lifestyle prediction, BACPHLIP[20] and DeePhage[36] were used.
- For phage host prediction, HostPhinder[37], VirHostMatcher[38], WIsH[38] and DeepHost[39] were used.
- (Phage integration site prediction is a novel task with no currently available tools.)

To evaluate if Graphage's performance could be merely due to its use of the k-mer pair distributions, we examine the performance of several conventional machine learning models when given the k-mer pair features as input. These models include SVM with linear function kernel and radial basis function kernel, k-nearest neighbors model, logistic regression model, AdaBoost model, decision tree model, and random forest model. More details on the parameters used in these methods are shown in Supplementary Method S1.1.

Performance evaluation metrics

The classification performances for all four tasks considered in this work are evaluated through accuracy, F1-score, and receiver operating characteristic (ROC) curve.

The F1-score is defined as

$$\text{F1-score} = \left(\frac{\text{Recall}^{-1} + \text{Precision}^{-1}}{2} \right)^{-1} \tag{5}$$

The metric is inapplicable for phage host prediction, which is a multi-class classification task. In this case, we measure weighted F1-score and macro F1-score instead.

$$\text{weighted F1-score} = \sum_{i=1}^{K} \frac{N_i}{N} \text{F1-score}_i \tag{6}$$

$$\text{macro F1-score} = \frac{1}{K} \sum_{i=1}^{K} \text{F1-score}_i \tag{7}$$

where K is the number of classes, N_i the number of samples in class i, and N is the total number of samples.

References

1. Rohwer, F. Global phage diversity. *Cell* **113**, 141 (2003).
2. Cobián Güemes, A. G. *et al.* Viruses as winners in the game of life. *Annu. Rev. Virol.* **3**, 197–214 (2016).
3. Dion, M. B., Oechslin, F. & Moineau, S. Phage diversity, genomics and phylogeny. *Nat. Rev. Microbiol.* **18**, 125–138 (2020).
4. Howard-Varona, C., Hargreaves, K. R., Abedon, S. T. & Sullivan, M. B. Lysogeny in nature: mechanisms, impact and ecology of temperate phages. *The ISME journal* **11**, 1511–1520 (2017).
5. Du Toit, A. Phage induction in different contexts. *Nat. Rev. Microbiol.* **17**, 126–127 (2019).
6. Fuhrman, J. A. Marine viruses and their biogeochemical and ecological effects. *Nature* **399**, 541–548 (1999).
7. Rodriguez-Valera, F. *et al.* Explaining microbial population genomics through phage predation. *Nat. Preced.* 1–1 (2009).
8. Brives, C. & Pourraz, J. Phage therapy as a potential solution in the fight against amr: obstacles and possible futures. *Palgrave Commun.* **6**, 1–11 (2020).
9. Gorski, A., Borysowski, J. & Miedzybrodzki, R. Phage therapy: Towards a successful clinical trial. *Antibiotics* **9**, 827 (2020).
10. Paez-Espino, D. *et al.* Uncovering Earth's virome. *Nature* **536**, 425–430 (2016).
11. Nayfach, S. *et al.* Metagenomic compendium of 189,680 DNA viruses from the human gut microbiome. *Nat. Microbiol.* **6**, 960–970 (2021).
12. Camarillo-Guerrero, L. F., Almeida, A., Rangel-Pineros, G., Finn, R. D. & Lawley, T. D. Massive expansion of human gut bacteriophage diversity. *Cell* **184**, 1098–1109 (2021).
13. Zhang, X. *et al.* Mining bacterial NGS data vastly expands the complete genomes of temperate phages. *bioRxiv* (2021).
14. Roux, S., Enault, F., Hurwitz, B. L. & Sullivan, M. B. VirSorter: mining viral signal from microbial genomic data. *PeerJ* **3**, e985 (2015).
15. Arndt, D. *et al.* Phaster: a better, faster version of the PHAST phage search tool. *Nucleic Acids Res.* **44**, W16–W21 (2016).
16. Song, W. *et al.* Prophage Hunter: an integrative hunting tool for active prophages. *Nucleic Acids Res.* **47**, W74–W80 (2019).
17. Tang, K. *et al.* Prophage Tracer: precisely tracing prophages in prokaryotic genomes using overlapping split-read alignment. *Nucleic Acids Res.* (2021).
18. Nayfach, S. *et al.* CheckV assesses the quality and completeness of metagenome-assembled viral genomes. *Nat. Biotechnol.* **39**, 578–585 (2021).
19. Edwards, R. A., McNair, K., Faust, K., Raes, J. & Dutilh, B. E. Computational approaches to predict bacteriophage–host relationships. *FEMS Microbiol. Rev.* **40**, 258–272 (2016).
20. Hockenberry, A. J. & Wilke, C. O. BACPHLIP: Predicting bacteriophage lifestyle from conserved protein domains. *PeerJ* **9**, e11396 (2021).
21. Stone, E., Campbell, K., Grant, I. & McAuliffe, O. Understanding and exploiting phage-host interactions. *Viruses* **11**, 567 (2019).
22. Lin, D. M., Koskella, B. & Lin, H. C. Phage therapy: An alternative to antibiotics in the age of multi-drug resistance. *World J Gastrointest Pharmacol Ther* **8**, 162 (2017).
23. Ahlgren, N. A., Ren, J., Lu, Y. Y., Fuhrman, J. A. & Sun, F. Alignment-free oligonucleotide frequency dissimilarity measure improves prediction of hosts from metagenomically-derived viral sequences. *Nucleic Acids Res.* **45**, 39–53 (2017).
24. Li, H. & Sun, F. Comparative studies of alignment, alignment-free and SVM based approaches for predicting the hosts of viruses based on viral sequences. *Sci. Reports* **8**, 1–9 (2018).
25. Song, K. Classifying the lifestyle of metagenomically-derived phages sequences using alignment-free methods. *Front. Microbiol.* **11**, 2865 (2020).
26. Huang, E.-S., Huong, S.-M., Tegtmeier, G. E. & Alford, C. Cytomegalovirus: genetic variation of viral genomes. *Annals New York Acad. Sci.* **354**, 332–346 (1980).
27. Zhang, Q., Shen, Z. & Huang, D.-S. Modeling in-vivo protein-DNA binding by combining multiple-instance learning with a hybrid deep neural network. *Sci. Reports* **9**, 1–12 (2019).

28. Li, M. & Zhang, W. PHIAF: prediction of phage-host interactions with GAN-based data augmentation and sequence-based feature fusion. *Briefings Bioinforma.* (2021).

29. Ilie, L. & Ilie, S. Multiple spaced seeds for homology search. *Bioinformatics* **23**, 2969–2977 (2007).

30. Battaglia, P. *et al.* Relational inductive biases, deep learning, and graph networks. *arXiv* (2018).

31. Kipf, T. N. & Welling, M. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)* (2017).

32. Brister, J. R., Ako-Adjei, D., Bao, Y. & Blinkova, O. NCBI viral genomes resource. *Nucleic Acids Res.* **43**, D571–D577 (2015).

33. Bi, D. *et al.* ICEberg: a web-based resource for integrative and conjugative elements found in Bacteria. *Nucleic Acids Res.* **40**, D621–D626 (2012).

34. Jiang, X., Hall, A. B., Xavier, R. J. & Alm, E. J. Comprehensive analysis of chromosomal mobile genetic elements in the gut microbiome reveals phylum-level niche-adaptive gene pools. *PloS One* **14**, e0223680 (2019).

35. Johnson, L. S., Eddy, S. R. & Portugaly, E. Hidden Markov model speed heuristic and iterative HMM search procedure. *BMC Bioinforma.* **11**, 1–8 (2010).

36. Wu, S. *et al.* Deephage: distinguishing virulent and temperate phage-derived sequences in metavirome data with a deep learning approach. *GigaScience* **10**, giab056 (2021).

37. Villarroel, J. *et al.* HostPhinder: a phage host prediction tool. *Viruses* **8**, 116 (2016).

38. Galiez, C., Siebert, M., Enault, F., Vincent, J. & Söding, J. WIsH: who is the host? Predicting prokaryotic hosts from metagenomic phage contigs. *Bioinformatics* **33**, 3113–3114 (2017).

39. Ruohan, W., Xianglilan, Z., Jianping, W. & Shuai Cheng, L. DeepHost: phage host prediction with convolutional neural network. *Briefings Bioinforma.* (2021).

40. McInnes, L., Healy, J. & Melville, J. UMAP: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

41. Bailey, T. L. STREME: accurate and versatile sequence motif discovery. *Bioinformatics* **37**, 2834–2840 (2021).

42. Gupta, S., Stamatoyannopoulos, J. A., Bailey, T. L. & Noble, W. S. Quantifying similarity between motifs. *Genome biology* **8**, 1–9 (2007).

43. Kılıç, S., White, E. R., Sagitova, D. M., Cornish, J. P. & Erill, I. CollecTF: a database of experimentally validated transcription factor-binding sites in bacteria. *Nucleic acids research* **42**, D156–D160 (2014).

44. Fornelos, N., Browning, D. F. & Butala, M. The use and abuse of LexA by mobile genetic elements. *Trends microbiology* **24**, 391–401 (2016).

45. Silpe, J. E. & Bassler, B. L. Phage-encoded LuxR-type receptors responsive to host-produced bacterial quorum-sensing autoinducers. *MBio* **10**, e00638–19 (2019).

46. Boyd, E. F., Davis, B. M. & Hochhut, B. Bacteriophage–bacteriophage interactions in the evolution of pathogenic bacteria. *Trends microbiology* **9**, 137–144 (2001).

47. Wang, C.-H. *et al.* Peroxide responsive regulator PerR of group A Streptococcus is required for the expression of phage-associated DNase Sda1 under oxidative stress. *PloS one* **8**, e81882 (2013).

48. Mikolov, T., Chen, K., Corrado, G. & Dean, J. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

49. Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S. & Dean, J. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 27*, 3111–3119 (2013).

50. Ng, P. dna2vec: Consistent vector representations of variable-length k-mers. *arXiv preprint arXiv:1701.06279* (2017).

51. Menegaux, R. & Vert, J.-P. Continuous embeddings of dna sequencing reads and application to metagenomics. *J. Comput. Biol.* **26**, 509–518 (2019).

52. Wang, C., Zhang, Y. & Han, S. Its2vec: fungal species identification using sequence embedding and random forest classification. *BioMed research international* **2020** (2020).

53. Zou, Q., Xing, P., Wei, L. & Liu, B. Gene2vec: gene subsequence embedding for prediction of mammalian N6-methyladenosine sites from mRNA. *RNA* **25**, 205–218 (2019).

54. Wahab, A., Tayara, H., Xuan, Z. & Chong, K. T. DNA sequences performs as natural language processing by exploiting deep learning algorithm for the identification of n4-methylcytosine. *Sci. Reports* **11**, 1–9 (2021).

55. Vang, Y. S. & Xie, X. HLA class I binding prediction via convolutional neural networks. *Bioinformatics* **33**, 2658–2665 (2017).

56. Phloyphisut, P., Pornputtapong, N., Sriswasdi, S. & Chuangsuwanich, E. MHCSeqNet: a deep neural network model for universal MHC binding prediction. *BMC Bioinforma.* **20**, 1–10 (2019).

57. Edgar, R. C. Search and clustering orders of magnitude faster than blast. *Bioinformatics* **26**, 2460–2461 (2010).

58. Ghodsi, M., Liu, B. & Pop, M. DNACLUST: accurate and efficient clustering of phylogenetic marker genes. *BMC Bioinforma.* **12**, 1–11 (2011).

59. Fu, L., Niu, B., Zhu, Z., Wu, S. & Li, W. CD-HIT: accelerated for clustering the next-generation sequencing data. *Bioinformatics* **28**, 3150–3152 (2012).

60. James, B. T., Luczak, B. B. & Girgis, H. Z. MeShClust: an intelligent tool for clustering DNA sequences. *Nucleic Acids Res.* **46**, e83–e83 (2018).

61. Seemann, T. Prokka: rapid prokaryotic genome annotation. *Bioinformatics* **30**, 2068–2069 (2014).

62. Mavrich, T. N. & Hatfull, G. F. Bacteriophage evolution differs by host, lifestyle and genome. *Nat. Microbiol.* **2**, 1–9 (2017).

63. Delcher, A. L., Bratke, K. A., Powers, E. C. & Salzberg, S. L. Identifying bacterial genes and endosymbiont DNA with Glimmer. *Bioinformatics* **23**, 673–679 (2007).

64. Fey, M. & Lenssen, J. E. Fast graph representation learning with PyTorch Geometric. *arXiv preprint arXiv:1903.02428* (2019).

65. Hamilton, W. L., Ying, R. & Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, 1025–1035 (2017).

66. Ma, Y. *et al.* Spectral-based graph convolutional network for directed graphs. *arXiv preprint arXiv:1907.08990* (2019).

67. Tong, Z., Liang, Y., Sun, C., Rosenblum, D. S. & Lim, A. Directed graph convolutional network. *arXiv preprint arXiv:2004.13970* (2020).

68. Zhang, X., Brugnone, N., Perlmutter, M. & Hirn, M. J. MagNet: A magnetic neural network for directed graphs. In *Advances in Neural Information Processing Systems 34* (2021).

69. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

70. Rubinstein, R. Y. & Kroese, D. P. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation, and machine learning*, vol. 133 (Springer, 2004).

71. Kingma, D. P. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

72. Paszke, A. *et al.* PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, 8026–8037 (2019).

73. Zielezinski, A., Vinga, S., Almeida, J. & Karlowski, W. M. Alignment-free sequence comparison: benefits, applications, and tools. *Genome Biol.* **18**, 1–17 (2017).

74. Řehůřek, R. & Sojka, P. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, 45–50 (ELRA, Valletta, Malta, 2010).

Acknowledgements

This work was supported by Strategic Interdisciplinary Research Grant (7020005).

Author contributions statement

RW designed the methods, implemented the networks, performed the experiments, and drafted the manuscript. YKN analyzed theoretical aspects of the ML models. XZ revised the manuscript. JW supervised this project and revised the manuscript. SCL conceived the topic, proposed the model, and revised the manuscript. All authors have read and approved of the final manuscript.

Conflict of interest

The authors declare that they have no competing interests.

Table

Method*	Accuracy	F1 score	AUC
Graphage (Ours)	**0.9340**	**0.8985**	**0.9805**
AF-based	0.8564	0.7629	-
word2vec	<u>0.8614</u>	<u>0.7868</u>	0.9105
Sequence-based	0.3317	0.4981	0.5487
GPD[12]	0.5924	0.6159	<u>0.9624</u>
SVM (linear kernel)	0.6766	0.6722	0.8845
SVM (rbf kernel)	0.3317	0.4981	0.8456
k-Nearest Neighbors	0.6683	0.6516	0.8977
Logistic Regression	0.3581	0.5057	0.8309
AdaBoost	0.5264	0.571	0.7968
Decision Tree	0.6007	0.5632	0.6448
Random Forest	0.3333	0.4987	0.6873

Table 1. Comparing Graphage with other models/tools on phage and ICE discrimination tasks. The best-performing model is in bold font, while the runner-up is underlined. AUC is inapplicable for the AF-based model since the model does not give probabilities.

Method	Accuracy	F1 score	AUC
Graphage (Ours)	**0.849±0.005**	**0.839±0.006**	**0.882±0.004**
AF-based	0.810±0.004	0.812±0.004	-
word2vec	0.791±0.007	0.779±0.010	0.844±0.008
Sequence-based	0.736±0.013	0.734±0.007	0.809±0.007
SVM (linear kernel)	0.575±0.071	0.659±0.029	0.681±0.062
SVM (rbf kernel)	0.500±0.007	0.665±0.006	0.773±0.009
K-Nearest Neighbors	0.690±0.006	0.741±0.006	0.801±0.006
Logistic Regression	0.782±0.007	0.782±0.006	0.858±0.005
AdaBoost	0.721±0.004	0.714±0.005	0.791±0.005
Decision Tree	0.699±0.008	0.702±0.009	0.698±0.008
Random Forest	0.802±0.008	0.786±0.006	0.874±0.002

Table 2. Comparing our graph model with other models/tools on phage integration site prediction tasks. The best performing model is in bold font, while the runner-up is underlined. Results are averaged over 10 random train-test splits.

Figure

Method	Accuracy	F1 score	AUC
Graphage (Ours)	**0.813±0.005**	**0.801±0.007**	**0.860±0.006**
AF-based	0.772±0.007	0.782±0.008	-
word2vec	0.769±0.006	0.763±0.006	0.836±0.006
Sequence-based	0.735±0.010	0.732±0.012	0.820±0.012
SVM (linear kernel)	0.623±0.053	0.660±0.044	0.694±0.058
SVM (rbf kernel)	0.687±0.038	0.727±0.007	0.800±0.005
K-Nearest Neighbors	0.588±0.031	0.690±0.006	0.681±0.008
Logistic Regression	0.761±0.023	0.756±0.006	0.832±0.004
AdaBoost	0.714±0.024	0.700±0.006	0.782±0.006
Decision Tree	0.685±0.007	0.690±0.007	0.685±0.007
Random Forest	0.785±0.021	0.767±0.031	0.855±0.007

Table 3. Comparing our graph model with other models/tools on bacterial integration site prediction tasks. The best performing model is in bold font, while the runner-up is underlined. Results are averaged over 10 random train-test splits.

Method	Accuracy	F1 score	AUC
Graphage (Ours)	**0.957±0.011**	**0.928±0.022**	**0.975±0.010**
AF-based	0.894±0.014	0.836±0.310	-
word2vec	0.939±0.011	0.898±0.021	0.959±0.009
Sequence-based	0.810±0.047	0.657±0.064	0.858±0.023
BACPHLIP[20]	0.932±0.007	0.888±0.016	0.956±0.009
DeePhage[36]	0.845±0.030	0.875±0.024	0.969±0.010
SVM (linear kernel)	0.914±0.024	0.858±0.044	0.958±0.015
SVM (rbf kernel)	0.819±0.023	0.642±0.048	0.860±0.038
K-Nearest Neighbors	0.875±0.023	0.803±0.044	0.926±0.016
Logistic Regression	0.890±0.027	0.809±0.054	0.950±0.019
AdaBoost	0.886±0.019	0.801±0.033	0.933±0.022
Decision Tree	0.818±0.023	0.704±0.039	0.789±0.034
Random Forest	0.890±0.018	0.805±0.040	0.948±0.019

Table 4. Comparing our graph model with other models/tools on phage lifestyle prediction tasks. The best performing model is in bold font, while the runner-up is underlined. Results are averaged over 10 random train-test splits.

Method	Accuracy	Weighted F1 score	Macro F1 score
Graphage (Ours)	**0.940±0.001**	**0.937±0.001**	**0.730±0.024**
AF-based	0.931±0.001	0.931±0.001	0.715±0.001
word2vec	0.746±0.032	0.725±0.033	0.358±0.014
Sequence-based	0.644±0.035	0.610±0.051	0.210±0.033
HostPhinder[37]	0.320±0.010	0.416±0.017	0.136±0.028
VirHostMatcher[38]	0.380±0.005	0.372±0.005	0.461±0.011
WIsH[38]	0.377±0.007	0.423±0.009	0.408±0.014
DeepHost[39]	0.885±0.017	0.870±0.023	0.398±0.104
SVM (linear kernel)	0.707±0.072	0.733±0.060	0.714±0.013
SVM (rbf kernel)	0.591±0.080	0.617±0.057	0.240±0.007
K-Nearest Neighbors	0.935±0.003	0.932±0.002	0.715±0.004
Logistic Regression	0.928±0.001	0.919±0.001	0.635±0.012
AdaBoost	0.588±0.006	0.460±0.010	0.030±0.006
Decision Tree	0.882±0.004	0.882±0.004	0.536±0.014
Random Forest	0.923±0.003	0.913±0.003	0.679±0.019

Table 5. Comparing our graph model with other models/tools on phage host species prediction tasks. The best performing model is in bold font, while the runner-up is underlined. Results are averaged over 5 random train-test splits.

Task	Dataset	Number of sequences	Number of classification
Phage and ICE discrimination	NCBI[32] and ICEBerg[33]	6,020	2
	ImmeDB[34]	606	2
	Total	6,626	2
Phage integration site prediction	integration sites on phages[13]	42,234	2
	integration sites on bacterial[13]	35,100	2
Phage lifestyle prediction	Phage evolution dataset[62]	1,551	2
Phage host prediction	GPD[12]	1,071	37
	Temperate phage dataset[13]	60,146	104
	Total	61,217	107

Table 6. Summary of benchmark datasets in this study.

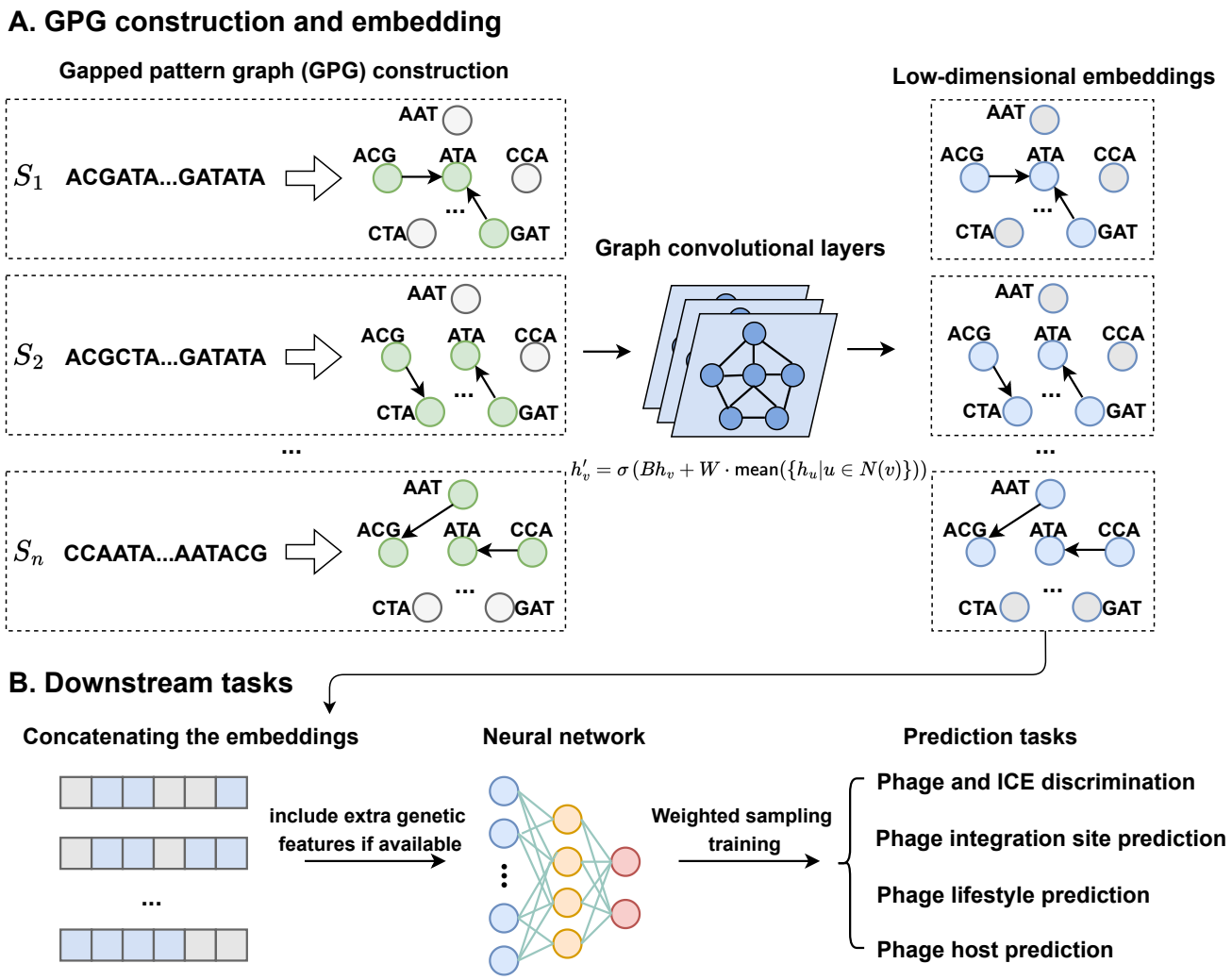

Figure 1. Overview of Graphage. A. A gapped pattern graph is constructed for each sequence and is embedded into latent spaces with a GCN. B. The embeddings are given as input to neural networks for four phage-related downstream tasks.

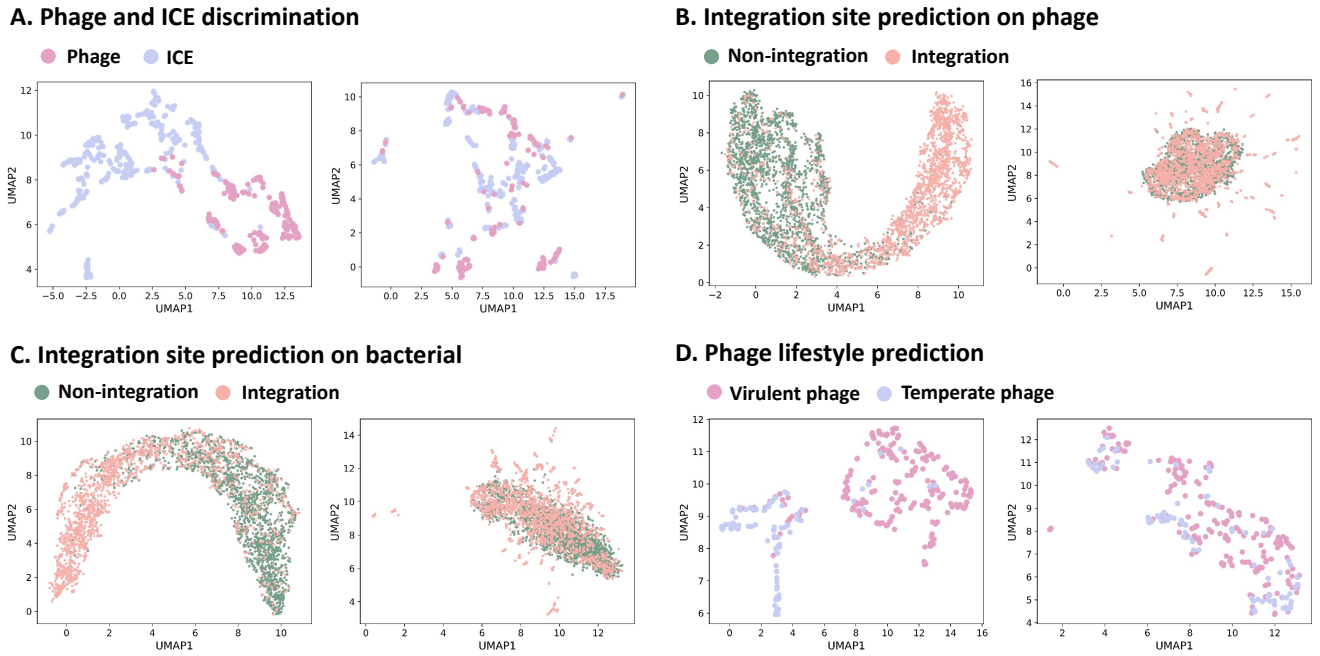

Figure 2. UMAP projections of the output from models with GCN and without GCN. For all subfigures A, B, C, and D, the latent features output by the models (prior to the final layer) with GCN are shown on the left, while that output by the models without GCN is shown on the right. Better separation between classes can be observed in the output of the models with GCN.

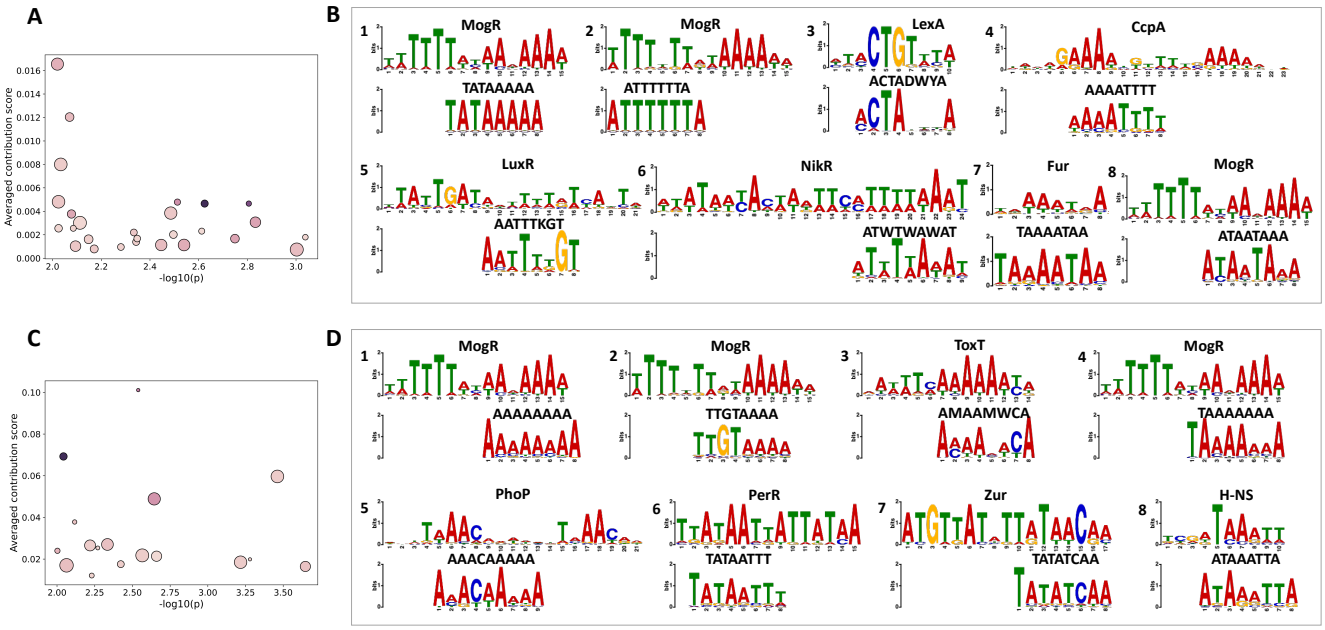

Figure 3. Motifs discovered in the integration site dataset match the TF-binding profiles in CollecTF. Some motifs enriched in the phage integration site (A and B) and bacterial integration site (C and D) also match the known motifs in the CollecTF dataset. In the scatter plots (A and C), we show the averaged contribution scores of the motifs calculated by Graphage and the alignment *p*-value reported by Tomtom, with the color suggesting the enrichment *e*-value and the size indicating the number of positive sequences matching the motif reported by STREME. The sequence logo alignment (B and D) between the motif in CollecTF (top) and the motif in the integration site (bottom) are shown for the top eight motifs according to the number of positive sequences.

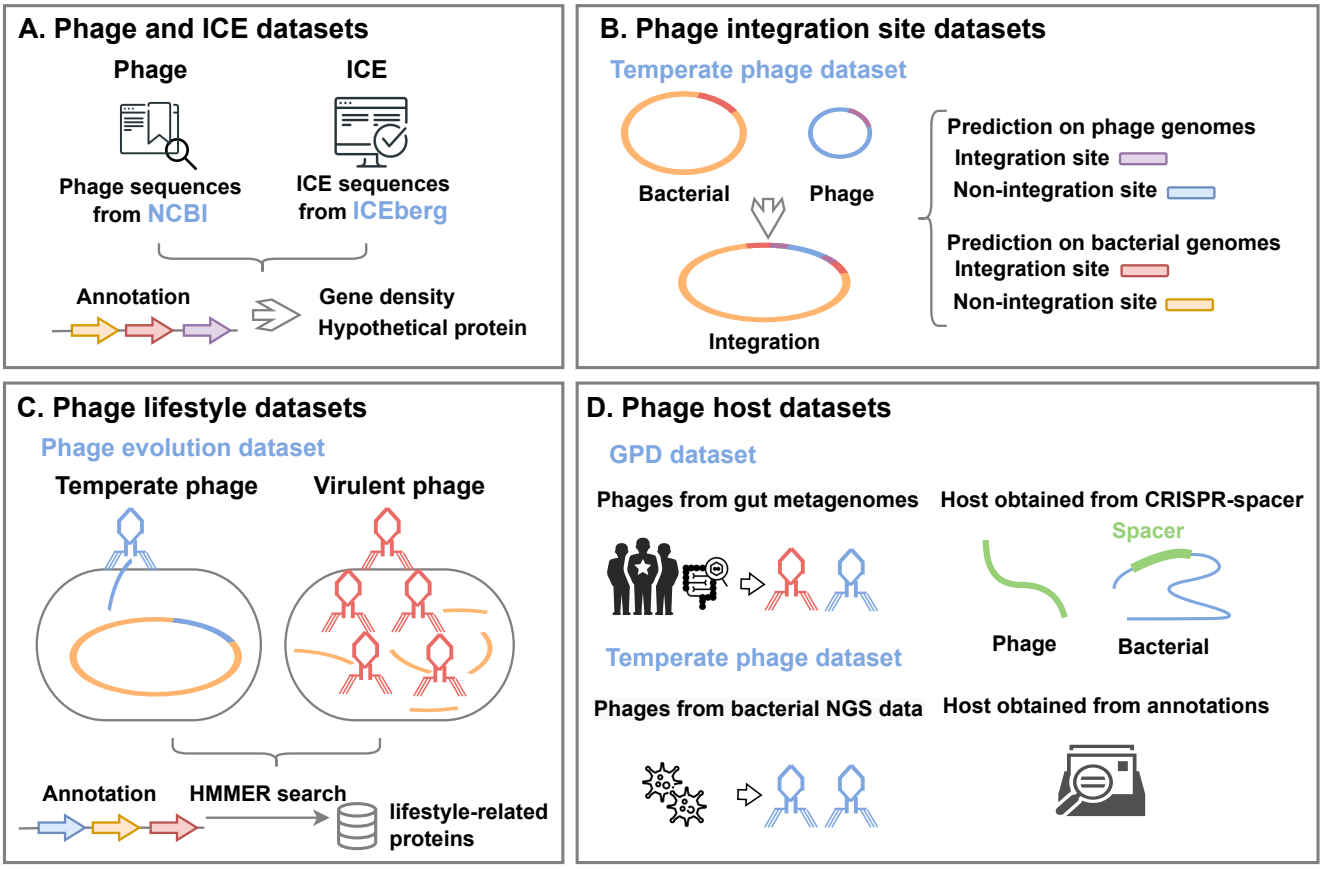

Figure 4. Four benchmark datasets are prepared, each for a phage-related task. A. Phage and ICE discrimination. B. Phage integration site prediction (on phage and bacterial genomes, respectively). C. Phage lifestyle prediction. D. Phage host prediction.